VA Informatics and Computing Infrastructure

Efficiently Accessing VINCI Data in R

Today's session begins at 2 PM ET

Download today's handout: https://tinyurl.com/y6lmgvhv

Today's caption link: https://tinyurl.com/yxhn29tx

We are streaming today's audio through your computer – please turn on your speakers or plug in a headset

OR

For the best audio quality have WebEx call you, or call in using:

(404) 397-1596, Access Code 199-727-8115, Password 3852

Using R Efficiently on VINCI

Andrew Redd, PhD. VINCI Helpdesk R Expert

Common Complaints

There are a few unavoidable issues I have found with VINCI

- Connection
- CAG - Citrix Access Gateway
- Layers and more layers.
- Servers restarting

- RStudio + Projects
- Packages
  - `VINCI`
  - Tidyverse
    - `dplyr`
    - `dbplyr` + `odbc` + `DBI`

RStudio®

on

accessible through:

- Standard applications directly or through
- remote desktop applications

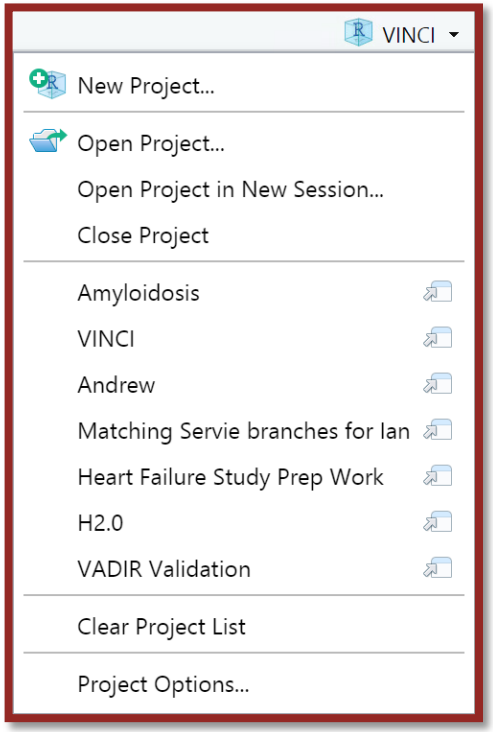
both are identical.

Projects

Leverage the project feature of RStudio

To create a new project either

- Click the new project button (R inside a box), next to the new document button
- Menu: File > New Project
- From the project menu (far right end of toolbar with dropdown icon) click new project.



Advantages of Projects

- Encapsulate & Organize files
- Manages working directory
  - Facilitates input and output
- Eases rerunning/rebuilding of scripts/reports

Packages on VINCI

- All packages on CRAN at the time R is updated are intalled with R.
- If you need updated packages or packages not on CRAN you must upload and install them
  - Download the windows binary of the package, and all dependencies not currently in VINCI.
  - Upload using the VINCI file upload.
  - Install into an R user or project library as a binary package.
    - you may need to specify the library with `.libPaths()` in the `.Rprofile` file or at the beginning of each file.

`VINCI` R Package

- `VINCI` is a context aware package that facilitates work in R on VINCI.
- More details throughout the presentation.
- Preview, working to get globally installed on VINCI

Tidyverse

If you are not versed in the tidyverse read up in:

- https://tidyverse.org
- R for Data Science

We will focus on the data flow portions today.

Data Tools - `dplyr`

`dplyr` abstracts data processes:

- Filter/Subset
- Select variables/features
- Join/Merge
- Transformations
- Renaming
- Etcetera

And implements these for R's traditional in-memory data.

dbplyr + odbc + DBI

- dbplyr implements the specifics of the dplyr operations for remote SQL tables.
- DBI implements a common data base interface.
  - connect/disconnect
  - list tables
  - write table
  - get query
  - etc.
- odbc implements the driver for specifics of ODBC connections.
  - Specific methods needed by DBI to construct appropriate queries.

Using the right tool for the job,

- R is great at complicated algorithms such as fitting models.
  - It's not great at managing lots of big datasets.
- SQL is great at managing datasets
  - It sucks at trying to fit complex models to the data.

Use the right tool for the job

What happens with `dbplyr`

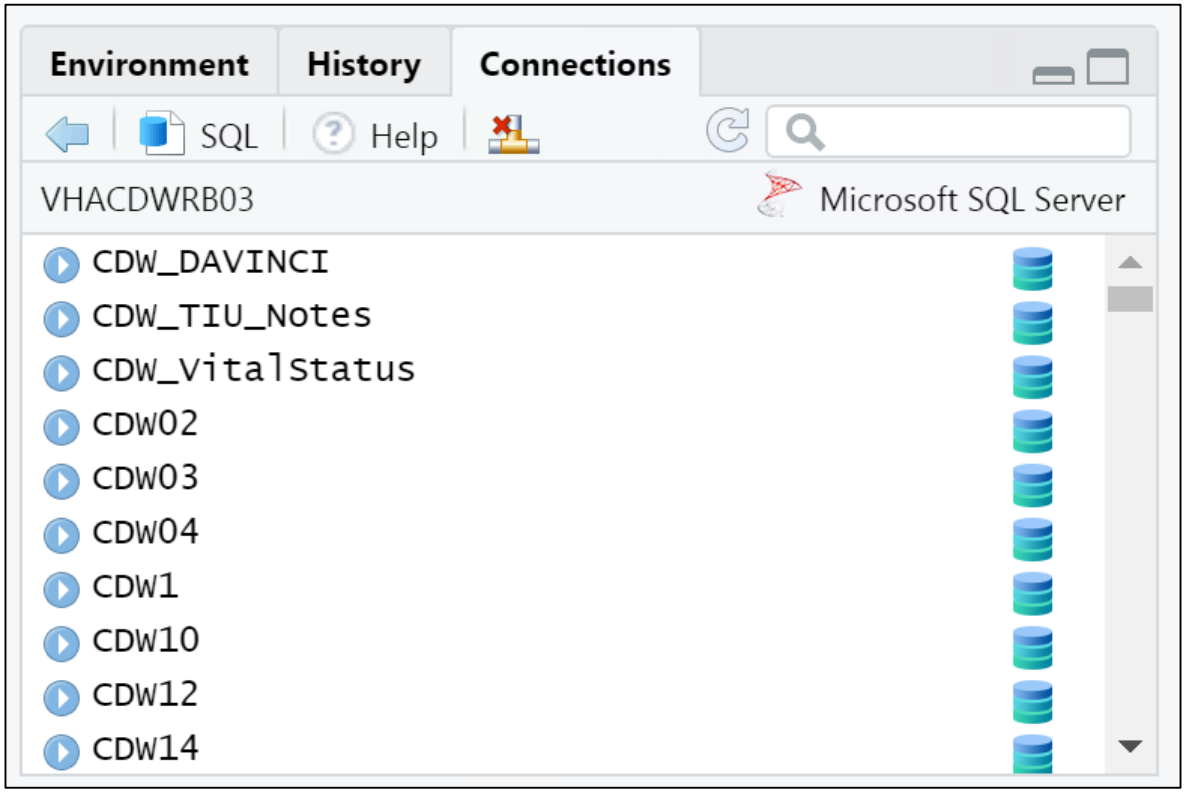
R is *lazy* and `dbplyr` takes this to an extreme:

- A table definition is created
  - including operations.
- Stays a definition until needed.
- Will combine with other definitions to create larger definitions.

Accessing Database Servers

- RStudio Data Connections
- Connection Strings
- `VINCI::VINCI_DB()`

RStudio® Data Connections



RStudio facilitates making connections through ODBC data source names when using the connections tab in the top right panel.

Connection Strings

ODBC connections can be made through "connection strings"

https://www.connectionstrings.com/sql-server/

May help in creating the connection string.

VINCI::VINCI_DB()

The easiest way to create a connection is through `VINCI_DB()`. It will

1. Examine the context of the RStudio project.
2. Attempt to find a database with the name matching the project context on the VINCI servers:
   - vhacdwrb01, vhacdwrb02, and vhacdwrb03
3. Construct the appropriate connection string
4. Establish an odbc connection to the database
5. Set the connection as the 'active' database connection and return the value back.

tbl & in_schema()

- Use `tbl()` to define a remote table.
- Use the `in_schema()` for specifying tables.

```
db <- VINCI_DB()
xwalk <- tbl(db, "Cohort_Crosswalk")
spatient <- tbl(db, in_schema("Src",
"SPatient_SPatient"))
```

- Nest `in_schema()` to access other databases on the same server.

```
dim.icd9 <- tbl(db, in_schema("CDWWork",
in_schema('Dim', 'ICDDescriptionVersion')))
```

Finishing computations

- `collect()` - to bring results back
- `compute()` - to perform query and save results in a table, temporary by default.
- `pull()` to perform a computation and bring back the resulting non-table result.

Caveat Emptor

- `dbplyr` does **not** perform any optimization on the SQL.
- It can create some very long queries.
- Complicated operation sets can easily overwhelm the SQL Server optimizer

Strategies

- Judicious use of `compute()`
  - especially If you reuse an intermediate result/query
- Cache Tables & data sets between sessions

Caching results

Saving intermediate results and final analysis datasets is not only good form it can substantially increase the time to get back into an analysis after a break.

VINCI::need_table()

The `need_table()` function will

1. Check if what you are looking for already exists in the workspace.
   - if found gently remind you of it,
   - but throw an error if it is not of an appropriate type.
2. Check the 'active' database connection for a matching table
   - assigns to the requested name a `tbl` to that table that if found.
3. Otherwise, if provided `code`, runs said code to create the table.

Example

For example running

```
need_table(Dim_ICD9)
```

Might produce something like

```
Using table '[Src].[Dim_ICD9]' found in database
ORD_Redd_202009010D
```

And upon second evaluation it would produce.

```
'Dim_ICD9' already a tbl_Microsoft SQL
Server/tbl_dbi/tbl_sql/tbl_lazy/tbl object
```

need_data()

`need_data()` is a traditional in-memory data analog of the `need_table()` function but it looks for data stored as `.RData` files in a data directory(ies)

A simple example might be

```
need_data(icd.labels, code={
    Dim_ICD9 %>% inner_join(ICDDescriptionVersion) %>%
    select(ICD9Code, ICD9Description) %>% distinct() %>%
    collect()
})
```

autosave()

`autosave()` is a simple utility to facilitate. It saves the object given in a file with the same name (with the `.RData` file extension) inside the of the `data` directory.

The `data` directory may be specified by

- The '`data.dir`' option.
- Creating a 'data' directory in the current working directory
- Setting a `DATA_DIR` environment variable.

A few helpful extras

Function	Description
nrow_sql()	gives the number of rows in an SQL table.
get_data_dir()	lists the data directories (first is the save location)
list_data()	lists data files in the data directory, see above.
is_key()	tests if a given set of variables forms a key (unique identifier) for the given table.
in_database()	replaces nested as_schema calls
sql_case()	alternate version of case_when that translates better with a default value.
find_tables_with()	searches database tables for tables will all the specified columns.
example:	find_tables_with('Sta3n', 'PatientSID')

Schema Specializations

- `info_schema()` creates `INFORMATION_SCHEMA` tibble views.
- Schema Manipulations:
  - `db_has_schema()`
  - `db_add_schema()`
  - `db_remove_schema()`
- `db_get_default_schema()` Retrieve the schema used if no schema is specified, usually the user name.
- `db_get_database()` get the database name
- `db_get_user()` get the logged in user.

SQL Server Improvements

This are transparent to the user but improve the functionality specifically with SQL Server.

- `db_has_table()`
- `db_save_query()`
- `sqlCreateTable()`

Bonus: Killing a frozen R session

Using R you have the ability to kill a frozen Rstudio or R session.

Follow these steps:

1. Load a new RStudio session if possible, if not start a Rgui session.
2. Get the active processes for the machine This takes a while.

```
wp <- wingui:::win_processes()
```

3. Find the problem process.

```
wp %>%
    filter( Image.Name == 'rsession.exe'
          , !is.na(User.Name))
```

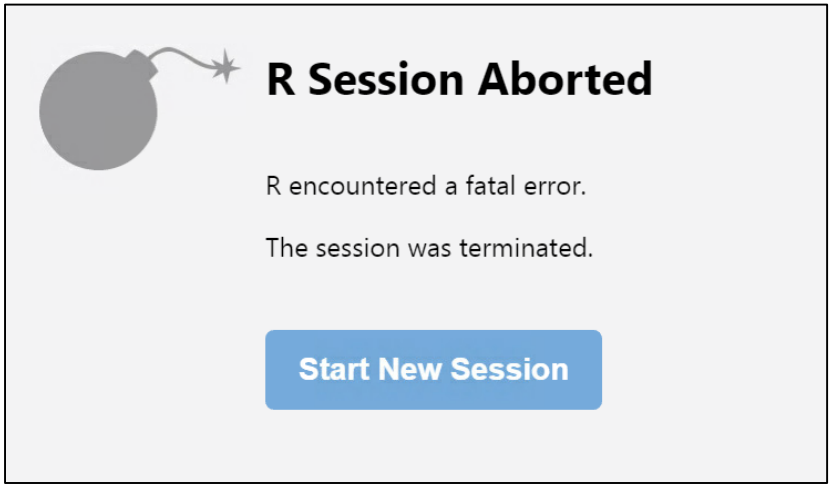
You should see at least two one for the one you are running in and the frozen one. It is up to you to figure out which one is the one to kill.

Bonus: Killing a frozen R session, Continued

1. Once you figure out which is the one to kill use `wingui:::win_kill()`

```
pid <- wp %>%
    filter( Image.Name == 'rsession.exe'
          , !is.na(User.Name)) %>%
    filter(PID == min(PID)) %>% pull(pid)
wingui:::win_kill(pid=pid)
```

If successful the frozen RStudio session will show a message



Bonus: Killing a frozen R session, notes

- You may need to kill RStudio itself (Replace `"rstudio.exe"` for `"rsession.exe"`)
- You could kill all processes of the same name, which is especially helpful if killing RStudio from Rgui

```
wingui:::win_kill(image="rstudio.exe")
```

- You cannot kill processes of other users.